Université Sultan Moulay Slimane
Ecole Nationale des Sciences Appliquées Khouribga

SUITE DE RAPPORT PROJET BIG DATA APRES SOUTENANCE

SYSTEME DE RECOMMANDATION DE FILMS

REDIGE PAR :

- SOKRI YASSER – N°41
- OUBADOU ABDELLAH – N°34

Pr. Soussi Nassima | Introduction au Big Data | 2021/2022

TABLE DES MATIERES

REMERCIEMENT :

- Mes vifs remerciements s'adressent à Madame Soussi Nassima, pour sa bienveillance et sa générosité et la grande patience dont il a su faire preuve durant le module d'Introduction au Big Data.
- Enfin, je tiens à remercier tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste projet, surtout mon collègue Rakib Ziad qui as créé l'image Docker que j'avais utilisé pour la réalisation de ce projet.

INTRODUCTION

- Apres avoir terminer la première phase vue dans le premier rapport, et générer le fichier final après une succession de jobs MapReduce pour implémenter l'algorithme de recommandation « content based » à l'aide de la méthode « TF IDF » pour avoir finalement le fichier résultat «de part-0000-1 » qui contient l'identifiant d'user et l'identifiant de film ainsi que la prédiction pour chaque couple « idMovie et idUser » dans le répertoire contentoutput9. Maintenant nous allons essayer de lier cette dernière avec un interface graphique en utilisant le Framework Bootstrap pour la partie frontend et spring boot pour le cote backend.

NB. : *Le TF-IDF est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus.*

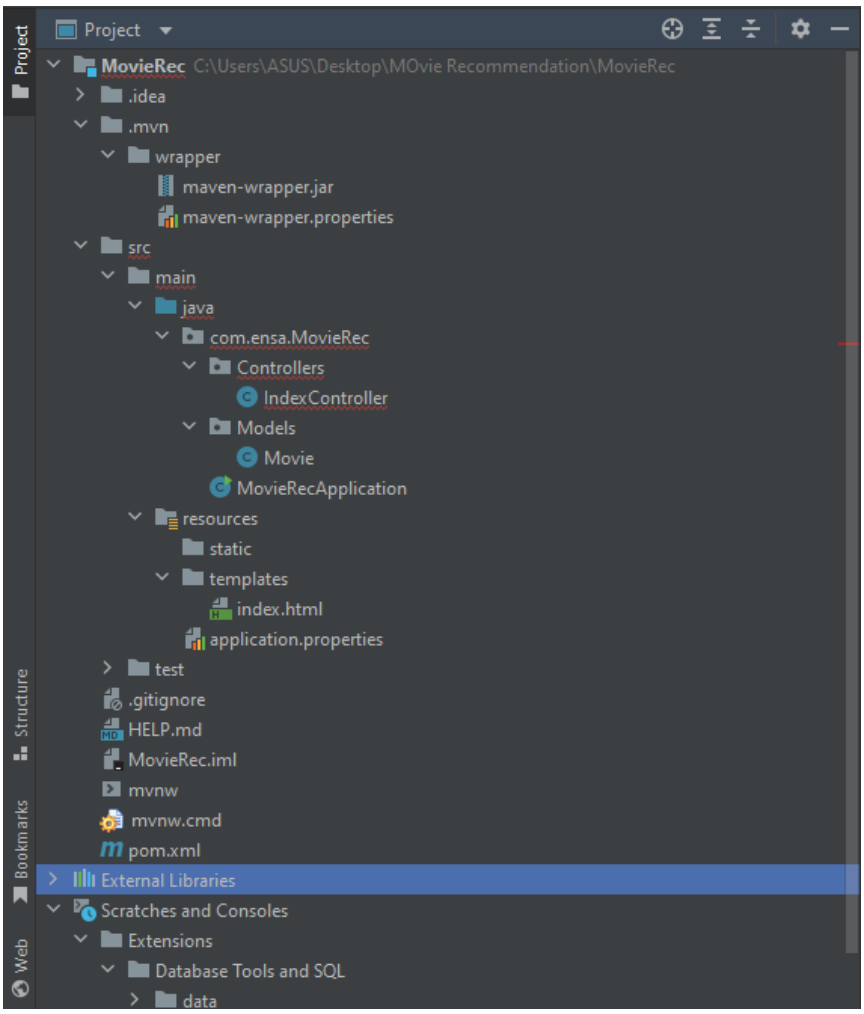
OUTILS

BOOTSTRAP

- Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

SPRING BOOT

- SpringBoot est un module récent de l'écosystème Spring. Il permet de simplifier le développement en Spring (configuration maven, programme principal, etc.).
- Les éléments nécessaires pour une application stand-alone en SpringBoot sont :
  - Un fichier pom.xml récupérant les éléments Spring Boot (pom parent)
  - Un fichier Main.java qui initialise le conteneur Spring
  - Des annotations dans les classes



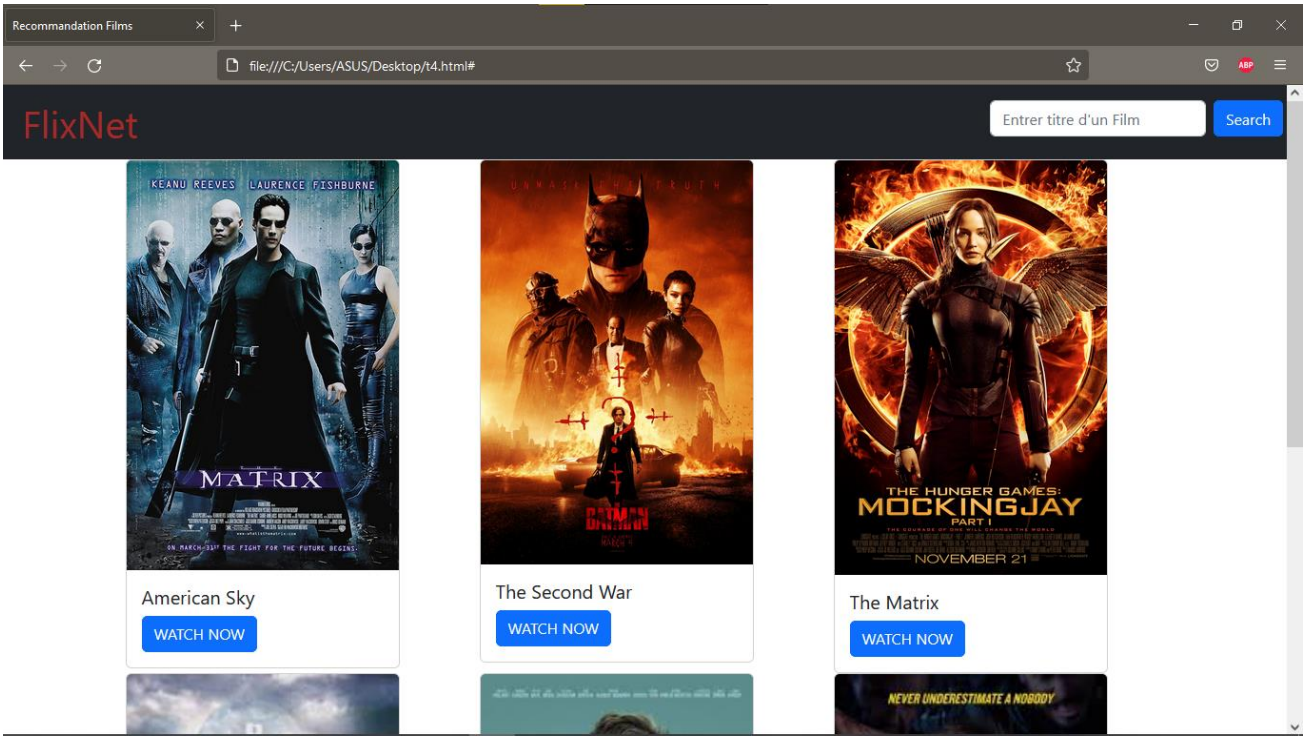
BEAN MOVIE:

Avant de démarrer le processus de récupération je vais créer un bean qui s'appelle movie, qui va être utiliser dans le model suite à l'architecture de JEE.

```java
package com.ensa.MovieRec.Models;

public class Movie implements Comparable<Movie> {
    private final int id;
    private final double score;
    private String name;
    private String imdbID;

    public Movie(int movieID, double score) {
        this.id = movieID;
        this.score = score;
    }

    public int getId() {
        return id;
    }

    public double getScore() {
        return score;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getImdbID() {
        return imdbID;
    }

    public void setImdbID(String imdbID) {
        this.imdbID = imdbID;
    }

    @Override
    public int compareTo(Movie M) {
        return Double.compare(getScore(), M.getScore());
    }
}
```

INTERFACE WEB :

Je vais, d'abord, essayer de créer une interface graphique en utilisant le Framework Bootstrap. Puis le rendre dynamique en récupérant les données résultantes de notre jobs MapReduce.



Code Source :

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Recommandation Films</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"
crossorigin="anonymous">
    <style>
        .nav{
          padding-left: 20px;
          color: brown;

        }

        .row{
          margin: 0;
          padding: 0;
        }

    </style>
```

```
  </head>
  <body>
    <nav class="navbar navbar-dark bg-dark">
      <h1 class="nav">FlixNet</h1>
      <form class="row g-2">
        <div class="col-auto">
          <label for="inputPassword2" class="visually-hidden"></label>
          <input type="password" class="form-control" id="inputPassword2"
placeholder="Entrer titre d'un Film">
        </div>
        <div class="col-auto">
          <button type="submit" class="btn btn-primary mb-3">Search</button>
        </div>
      </form>
    </nav>
  <div class="container">
    <div class="row">
      <div class="col" th:each="movie: ${movieList}" >
        <div class="card" style="width: 18rem;">
          <img th:src="'https://img.omdbapi.com/?apikey=df97dc6a&i='+
${movie.imdbID}" class="card-img-top" alt="...">
          <div class="card-body">
            <h5 class="card-title" th:text="${movie.name}"></h5>
            <a href="#" class="btn btn-primary">WATCH NOW</a>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

MANIPULATION DES FICHIER A LAIDE DE L'API JAVA

- Ensuite, à travers un contrôleur je vais essayer de récupérer directement les recommandations de l'HDFS en manipulant les fichiers générés à l'aide du paradigme MapReduce en utilisant l'API Java comme ce qu'on a vu durant les tps. Premièrement il faut importer les classes et les modules nécessaires pour notre traitement, ainsi que les classes de spring nécessaires.
- Lire du fichier résultat part-oooo… les Ids de l'utilisateur, de films, et la prédiction. Ensuite, aller vers le fichier movies pour récupérer le titre et puis aller vers le fichier links pour récupérer le lien de l'image pour la générer dans l'interface graphique de notre application.

```java
package com.ensa.MovieRec.Controllers;

import com.ensa.MovieRec.Models.Movie;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;
import java.util.stream.Collectors;

@Controller
public class IndexController {
    @GetMapping("/")
   String index(Model model) {
      HashMap<Integer, Movie> movieDict = new HashMap<>();
      try {
         final int selectedID = 1;
         Path pt = new Path("contentoutput9/part-r-00000");
         FileSystem fs = FileSystem.get(new Configuration());
         BufferedReader br = new BufferedReader(new
InputStreamReader(fs.open(pt)));
         String line = br.readLine();
         while (line != null) {
            String[] fields = line.split("\t");
            int userID = Integer.parseInt(fields[0]);
            if (userID > selectedID) {
               break;
            }
            else if (userID < selectedID)
            {
               continue;
            }
            String[] movieScore = fields[1].split(",");
            int movieID = Integer.parseInt(movieScore[0]);
            Movie movie = new Movie(movieID, Double.parseDouble(movieScore[1]));
            movieDict.put(movieID, movie);
            line = br.readLine();
         }
         pt = new Path("movies.csv");
         br = new BufferedReader(new InputStreamReader(fs.open(pt)));
         line = br.readLine();
         while (line != null) {
            String[] fields = line.split(",");
            int movieID = Integer.parseInt(fields[0]);
            String movieName = fields[1];
            Movie movie = movieDict.get(movieID);
            if(movie != null) movie.setName(movieName);
            line = br.readLine();
         }
         pt = new Path("links.csv");

         br = new BufferedReader(new InputStreamReader(fs.open(pt)));
         line = br.readLine();
```

```
        while (line != null) {
            String[] fields = line.split(",");
            int movieID = Integer.parseInt(fields[0]);
            String imdbID = "tt" + fields[1];
            Movie movie = movieDict.get(movieID);
            if(movie != null) movie.setImdbID(imdbID);
            line = br.readLine();
        }
    } catch (Exception e){
        e.printStackTrace();
    }
    List<Movie> movieList = new ArrayList<>(movieDict.values());
    Collections.sort(movieList);
    Collections.reverse(movieList);
    for (Movie movie: movieList) {
        System.out.println(movie.getName() +" "+movie.getScore());
    }

    model.addAttribute("movieList", movieList);
    return "index";
  }
}
```

REMARQUE :

- Afin de mieux clarifier les étapes ci-dessous, je vous laisse avec d'autres pièces jointes une vidéo de démonstration de l'exécution de cette petite application, ainsi que le code source utilisé.

CONCLUSION :

- Ce mini-projet présente un système de recommandation basé sur le contenu.
- Le système de recommandation est devenu de plus en plus important en raison de la surcharge d'informations. Pour le système de recommandation basé sur le contenu en particulier, nous avons essayé de trouver un moyen pour recommander un film en utilisant le Framework MapReduce.

- Pour les problèmes qui s'est posé, nous utilisons d'abord un algorithme de recommandation basé sur le contenu, ce qui signifie qu'il n'y a pas des informations de démarrage pour utiliser l'algorithme pour la première fois.